REMARKS/ARGUMENTS

~~Claims 1-30 are pending in the present application. Claims 8 and 15-16 were amended. Reconsideration of the claims is respectfully requested.~~

The application identified above has been amended pursuant to a Request for Continued Examination for such application. Claims 1-7, 16-23, and 31-40 are pending in the present application. Claims 8-15 and 24-30 were canceled. Claims 1-8, 16-19 and 22-23 were amended, and Claims 31-40 were added. Reconsideration of the claims is respectfully requested.

Applicants filed a document entitled Response to Office Action regarding the above application with the United States Patent and Trademark Office (USPTO), on September 29, 2006. Comments and remarks set forth in this document (hereinafter "Response") are incorporated herein by reference.

I. 35 U.S.C. § 112, Second Paragraph

In a Final Office Action mailed December 13, 2006, (hereinafter "Final Office Action"), the Examiner rejected Claims 8-15 as being indefinite. These claims have been canceled. Accordingly, this rejection has now been overcome.

II. 35 U.S.C. § 103, Obviousness

In the Final Office Action the [[The]] Examiner has rejected Claims 1-7 and 24-30 under 35 U.S.C. § 103(a) as being unpatentable over a document entitled "An Evaluation of Speculative Instruction Execution on Simultaneous Multithreaded Processors," a publication of *Swanson* et al. (hereinafter "*Swanson*"), in view of U.S. Patent Publication No. 2003/0182536, to *Teruyama* (hereinafter "*Teruyama*"). Claims 16-23 were rejected as being unpatentable over U.S. Patent No. 5,555,432 to *Hinton et al* (hereinafter "*Hinton*"), in view of *Swanson*. These rejections are respectfully traversed.

III. 35 U.S.C. § 102, Anticipation

The Examiner has rejected Claims 8-15 under 35 U.S.C. § 102 as being anticipated by *Hinton*. These claims have been canceled. Accordingly this rejection has now been overcome.

IV. <u>Teachings of Applicants</u>

Applicants' invention relates to a method for issuing instructions in a multi-threaded microprocessor or the like that includes at least one multi-stage pipeline. As taught by Applicants, and as is known by those of skill in the art, a pipeline processor has multiple stages that each instruction must traverse during processing, after it has been issued. At paragraphs [0004] and [0018], Applicants teach that "issue", as defined by their application, "means forwarding an instruction to the pipeline for processing".

In making their invention, Applicants were concerned about situations in which processed instructions caused stalls in a pipeline, which could result in the entire pipeline becoming stalled. Applicants recognized that a major cause of stalling is dependency, wherein processing of an instruction through a pipeline requires that a result must have been successfully achieved by a previous or prerequisite instruction. Moreover, Applicants recognized that in a multi-threaded processor, as used with their invention, instructions are received simultaneously from different threads, as a set. Therefore, stalling could also occur as a result of conflicts over a resource that was shared by different instructions of the set. Thus, to significantly diminish stalling, Applicants position an instruction issue logic component between instruction input buffers and the pipeline processor. The instruction issue logic determines a critical range or distance for a dependent instruction, wherein critical distance is the number of pipeline stages between a stage when the dependent instruction will need a result, and the stage at which the result is made available by a prerequisite instruction. The instruction issue logic also resolves conflicts between instructions of the set over shared resources. Based on the critical distance and other information, the instruction issue logic determines a probability that an instruction will complete all pipeline stages. The instruction issue logic then selects the instruction from the received set that is <u>least likely</u> to cause a stall, and issues the selected instruction for processing, provided it has a probability that is above a predetermined threshold.

The above teachings of Applicants are set forth in the specification, such as at Figure 5 and portions of paragraphs [0018], [0020] and [0021], as shown below:

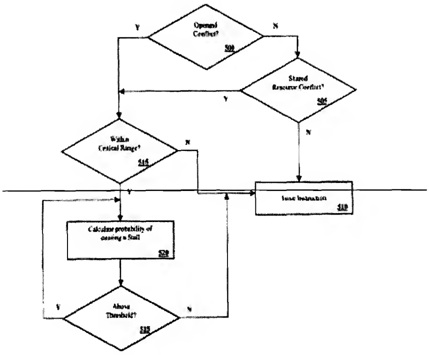

Figure 5

<u>One instruction from each column of buffers enters instruction issue logic 200 where dependency problems for any instruction of that set are identified.</u> If an instruction is found to require an operand that will likely not be available when the instruction needs it, then the thread that the instruction came from is withdrawn from the pool of candidates for issuing, wherein issuing means forwarding an instruction to the pipeline for processing. <u>The thread of instructions that is least likely to cause a stall in the pipeline is then issued.</u> That is not to say that the entire thread is guaranteed to traverse the pipeline without interruption. <u>The instruction issue logic 200 evaluates a probability of causing a stall on every clock cycle in the preferred embodiment.</u> Thus, if an instruction from a thread that is currently being processed is found to have a high probability of causing a stall, that thread will be delayed and an instruction from another thread will issue. When an instruction is issued, it enters the first stage "A" of the shared pipeline stages 210. **[from [0018], page 7] (emphasis added)**

In step 420 instructions with potential conflicts are flagged so that the instruction issue logic can calculate a probability of causing a stall for those instructions.

In step 425 the likelihood of causing a stall is determined. If the likelihood of causing a stall is below a threshold level, such as 50% for example, then the instruction is allowed to issue, step 435. **[from [0020], pp. 9-10]**

> **[0021]** Figure 5 is flow chart showing the steps followed once an instruction has been flagged as having a potential conflict. In step 500 it is determined whether or not the potential conflict is an operand conflict. If there is no operand conflict, then it is determined whether or not there is a shared resource conflict, step 505. If there is not a shared resource conflict then the instruction is allowed to issue, step 510. If however, in step 500 or 505, it is determined that there is a conflict, then the instruction issue logic determines whether or not the two conflicting instructions are within a critical range of each other, step 515. Using the example of Figure 2, which had a critical range for operand conflicts of four clock cycles, if the dependent instruction is four or more pipeline stages behind the instruction upon which it depends ("independent" instruction), then the dependent instruction is allowed to issue, step 510. <u>If however, the dependent instruction is less than four stages behind the "independent" instruction, then the probability of causing a stall must be calculated.</u> The critical range for shared resource conflicts will be one or more clock cycles less than the critical range for operand conflicts, depending upon the resource involved. In step 520 the probability of causing a stall in the pipeline is determined based on the critical range and the current contents of the pipeline. In step 525 the probability is compared to a threshold value, which can vary. **(emphasis added)**

Claim 1 recites a useful embodiment of Applicants' invention, and reads as follows:

> 1. (Currently Amended) A method for issuing instructions in a multithreaded computer processor, the method comprising the steps of:
>
> receiving a set of computer instructions in an instruction issue logic, wherein each instruction of said set comprises one instruction from each of a plurality of independent instruction threads;
>
> identifying as dependent instructions those received instructions that require a result from a prerequisite instruction;
>
> determining a probability for each received instruction that the instruction will complete all stages of a multi-stage instruction pipeline of the processor without causing a stall;
>
> selecting the received instruction of the set that is least likely to cause a stall in the multi-stage pipeline; and
>
> issuing, the selected instruction into the pipeline for processing, from the instruction issue logic, when the probability for the selected instruction is above a predetermined threshold.

V. <u>Claim 1 Distinguishes over Cited References</u>

In order to establish a *prima facie* case of obviousness under 35 U.S.C. § 103, **MPEP § 2143** requires that there must be some suggestion or motivation in the prior art to modify the reference or to combine reference teachings as proposed, and the prior art or combined references must teach or suggest <u>all</u> of the claim litigations. The suggestion to make the claim combination must be found in the prior art, not in the Applicants' disclosure. *In re Vacek*, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991). Moreover, in accordance with **MPEP § 2142.02**, each prior art reference must be considered in its <u>entirety</u>, i.e., as a whole, including portions that would lead away from the claimed invention. *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 220 U.S.P.Q. 303 (Fed. Cir. 1983). A third essential requirement for establishing a *prima facie* case, set forth in **MPEP § 2143.01**, is that the proposed modification cannot render the prior art unsatisfactory for its intended purpose.

In the present case, not all of the features of the claimed invention have been properly considered, and the teachings of the references themselves do not teach or suggest the claimed subject matter to a person of ordinary skill in the art. For example, no combination of *Swanson*, *Teruyama*, and/or *Hinton* teaches or suggests, in the over-all combination of Claim 1, either of the following Claim1 features:

(1) The step of determining a probability for each received instruction that the instruction will complete all stages of the multi-stage instruction pipeline without causing a stall (hereinafter "Step (1)").

(2) The step of selecting the received instruction of the set that is least likely to cause a stall in the multi-stage pipeline (hereinafter "Step (2)").

Applicants consider that none of the cited references discloses Step (1) of Claim 1, that is, determining a probability for each instruction received at the instruction issue logic that the instruction will complete all pipeline stages without causing a stall. As for *Swanson*, the *Swanson* reference evaluates the use of speculative techniques to execute instructions on simultaneous multithreaded (SMT) processors, which may use pipeline processing. However, *Swanson* emphasizes repeatedly that its disclosure is directed only to processing errors caused by wrong path speculative instructions. This is demonstrated, for example, by repeated reference to paths and branches in *Swanson*, such as in sections thereof referenced in the Final Office Action as pertaining to Claim 1. In regard to wrong path errors, processing of an instruction follows a path that can encounter one or more branches. Accordingly, *Swanson* discloses the use of confidence levels or confidence estimators to predict the probability that processing will follow the right path or branch, rather than a wrong path or unexpected branch.

It is readily apparent that since *Swanson* is exclusively concerned with wrong path errors, it provides no teaching in regard to processing instructions that include dependent instructions, or to problems of stalling when processing such instructions. Thus, *Swanson* clearly fails to show or suggest important teachings of Applicants' claims, including the recitation of Step (1) of Claim 1. From sections of the *Teruyama* reference such as the abstract, paragraph [0014] and claim 1 thereof, it is seen that *Teruyama* discloses an arrangement wherein a first circuit detects direct dependencies between issued instructions and a load instruction in each stage of a pipeline processor. In addition, a second circuit detects indirect dependencies between issued instructions and a load instruction which has cache-missed. As taught at paragraph [0041], a cache miss occurs when data is not loaded into a cache as required by a load instruction. When a data cache miss occurs, load store unit 18b outputs a cache miss signal, which is supplied to DLC 16. Paragraph [0124] discloses that DLC 16 comprises first and second detecting circuits 16a and 16b, respectively.

It is readily apparent that these teachings do not disclose determining a probability that dependent or other instructions will not cause a stall in a pipeline processor. Such teachings would not be necessary in the *Teruyama* arrangement. Thus, *Teruyama* likewise does not disclose Step (1) of Claim 1.

The *Hinton* reference, of course, is directed to a single threaded processor, and not to the multithreaded processor of Applicants' invention. Principal teachings of *Hinton* are set forth, for example, at the abstract, col. 11, lines 14-49 and col. 14, lines 42-64 thereof, as well as at Figure 5. These sections are as follows:

ABSTRACT

An out-of-order execution processor comprising an execution unit, a storage unit and a scheduler is disclosed. The storage unit stores instructions awaiting availability of resources required for execution. The scheduler periodically determines whether resources required for executing each instruction are available, and if so, dispatches that instruction to the execution unit. The execution unit indicates future availability of hardware resources such as functional units and write back ports a number of clock cycles before actual availability of the hardware resources. The scheduler determines availability of resources required for execution of an instruction based on the indication of future availability of the hardware resources, and dispatched the instruction for execution. The out-of-order execution processor also includes means to determine future completion of execution of source instructions a number of clock cycles before actual completion of execution. The scheduler dispatches for execution a data-dependent instruction that requires an execution result of one of such source instructions for an operand. Once the execution result of the source instruction is available, a bypass multiplexor bypasses the execution result into the dispatched data-dependent instruction. The bypass multiplexor sends the data dependent instruction with fully assembled operands to the execution unit for execution. **(emphasis added)**

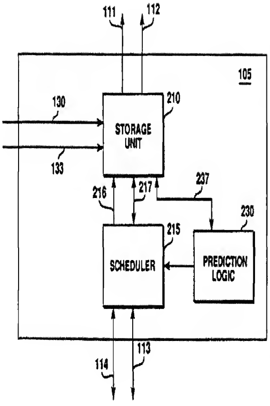

FIGURE 5

FIG. 5 illustrates the reservation station 105 for one embodiment. The reservation station 105 comprises a storage unit 210, a scheduler 215, and prediction logic 230. The storage unit 210 stores the instructions received from the front end unit 103 over the instruction issue bus 133. The scheduler 215 determines whether each stored instruction is ready for execution and dispatches the ready instructions in the storage unit 210 to one of the bypass multiplexors 115, 116 by sending a dispatch signal over a dispatch signal bus 216. The prediction logic 230 predicts availability of execution results of instructions that are scheduled for execution.

The prediction logic 230 determines future availability of execution results by examining ready instructions, and sends a one clock preliminary valid signal to the storage unit 210 accordingly. The prediction logic 230 examines the operation code of ready instructions that are ready for dispatch in each clock cycle. If the execution time of the ready instruction is one clock cycle ("one clock instruction"), the prediction logic 230 sends a one clock preliminary valid signal with the result tag of the one clock instruction to the storage unit 210 over the bus 231. The one clock preliminary valid signal indicates that the corresponding execution result will be written back on the write back bus 109 or 110 in two clock cycles.

The one clock preliminary valid signal causes the scheduler 215 to schedule a data dependent instruction for execution if all the resources, other than the operand corresponding to the execution result, are available. The scheduler 215 determines a data dependent instruction is ready one clock cycle (i.e. back-to-back scheduling) after determining the one-clock-instruction is ready. The one clock lag permits the execution result of the one-clock-instruction to be available at the bypass multiplexors 115, 116 during the same clock cycle as when the data dependent instruction is received at the bypass multiplexor 115, 116.

[col. 11, lines 14-49]

FIG. 7 is a flowchart illustrating high level steps in scheduling and executing instructions in processor 10 for one embodiment. In step 401 (*a*), the scheduler 215 determines whether each instruction will have operands available by the time the instruction arrives at the execution unit. The scheduler 215 determines that an operand will be available either if the operand is available or if the future availability of execution result corresponding to the operand has been predicted. The scheduler 215 determines that operands 1 and 2 are available if the valid fields 342 and 352 respectively are set to one.

The scheduler 215 determines that operand 1 will be available by the time the instruction is received at the bypass multiplexor 115 or 116 if the 2 clock preliminary valid bit 344 or 1-clock preliminary valid bit 345 is set to 1. A value of one in the 2-clock preliminary valid bit 344 indicates that one of the execution units 107, 108 has sent a 2-clock preliminary valid signal indicating the future availability of the execution result that corresponds to the operand. A value of one in the 1-clock preliminary valid field 345 indicates that the prediction logic 230 has sent a 1-clock preliminary valid signal indicating the future availability of execution result corresponding to operand 1.

Similarly, the scheduler 215 determines that operand 2 corresponding to entry 300 will be available by the time the instruction is received at the bypass multiplexor 115 or 116

[col. 14, lines 42-64]

The *Hinton* abstract teaches that instructions are executed based on indications of future availability. Thus, if it is expected that resources required for executing an instruction will be available in the future, *Hinton* will dispatch the instruction to the execution unit. This teaching is reinforced at col. 11, lines 25-49 of *Hinton*, which describes a prediction logic 230 thereof. Prediction logic 230 determines future availability of an execution result, and responds by sending a one clock preliminary valid signal. This signal in turn causes a scheduler 215 to schedule a dependent instruction for execution, if all needed resources, other than an operand corresponding to the execution result, are available. Thus, *Hinton* teaches that prediction of future availability of an execution result is to be treated as definite or as a certainty. These teachings of *Hinton* clearly direct away from using a probability, as required by Step (1) of Claim 1, in connection with execution of a dependent instruction. Probability is meaningful only where there is some recognition of uncertainty.

Hinton further emphasizes these teachings such as at col. 14, lines 46-53. Therein, *Hinton* indicates that a prediction of future availability of an operand to be treated with the same certainty as an operand that presently is available. However, such teachings are clearly incompatible with the recitation of Step (1) of Claim 1. In reciting the determining of a probability for an instruction, Step (1) teaches that some uncertainty, as to the progress of the instruction through the pipeline, must be provided for.

Applicants consider that none of the cited references discloses Step (2) of Claim 1, that is, selecting the received instruction of the set that is least likely to cause a stall in the multi-stage pipeline. The *Hinton* reference would have no use for such step, since the teachings of *Hinton* are directed to a single threaded processor, which does not receive a set of instructions, as does the multi-threaded processor of Applicants.

Additionally, other features of Claim 1 are considered to distinguish over the cited art. For example, the step of issuing the selected instruction into the pipeline for processing, from the instruction issue logic, is considered to distinguish over *Swanson* and *Teruyama*, for the same reasons given in support for Feature (2) of Claim 1, in Applicants' Response to Office Action filed September 29, 2007.

Independent Claims 16 and 34 respectively incorporate patentable subject matter of Claim 1, and are each considered to distinguish over the art, including the cited references, for at least the same reasons given in support for Claim 1.

In order to reject Claim 1, 16 or 34 for obviousness under 35 U.S.C. § 103, by any combination of *Hinton* and/or *Teruyama*, the Examiner must first demonstrate that there is some basis or motivation <u>in the prior art</u> for making the proposed combination. As discussed by Applicants in their prior Response, courts <u>continue</u> to hold that it is <u>absolutely essential</u> to provide such prior art motivation, in order to establish a *prima facie* case of obviousness by combining references. However, in the Final Office Action, no basis was provided for combining cited references to reject Claim 1 or 16, except for unsupported reasons as to why the combinations would be obvious to a person of ordinary skill in the art. Applicants consider that such unsupported reasons are clearly insufficient to meet the motivational requirements of 35 U.S.C. § 103. Applicants believe that they are entitled to <u>challenge</u> the alleged basis or motivation for combining references against their claims. To do this, Applicants must have a citation to a prior art reference that teaches the <u>reason</u> for the combination. This is necessary to provide a clear and complete picture of what was <u>actually done previously</u> by one of skill in the art, not an opinion as to what one of skill in the art might do.

VI. <u>Claim 31 Distinguishes over Cited References</u>

Independent Claim 31 is considered to distinguish over the cited references for at least some of the reasons given in support for Claim 1. In addition, Claim 31 distinguishes over such references in reciting the calculating and determining steps thereof, in combination. Such steps collectively include the tasks of (1) calculating a critical distance, comprising the number of stages between a stage when a selected dependent instruction will meet a given result, and a stage when the result will be available; (2) determining whether the selected instruction is within the critical distance; and (3) if the selected instruction is within the critical distance, determining a probability that the selected instruction will complete all stages of the pipeline without causing a stall. Applicants consider that neither *Swanson, Teruyama,* nor *Hinton*, nor any combination thereof, discloses this recited combination of tasks, in the overall combination of Claim 31. Clearly, none of such references shows determining a probability for an instruction, based on <u>whether</u> the instruction was within the critical distance of Claim 31.

VII. <u>Remaining Claims Distinguish over Cited References</u>

Claims 2-7, 17-23, 32-33 and 35-40 depend on independent Claims 1, 16, 31 and 34, respectively, and are each considered to distinguish over the art for the same reasons given in support thereof.

Claim 2, 17, and 35 are additionally considered to distinguish over the art in reciting the feature of determining whether there is a shared resource conflict between two or more of the received instructions of the set. Neither *Swanson*, *Hinton*, nor *Teruyama*, nor any combination thereof, shows or suggests this feature, particularly in regard to shared resource conflicts between the instructions of the same received set.

Claims 4, 19, and 37 are additionally considered to distinguish over the art in reciting the feature that the probability for each received instruction is expressed as a percentage value, and the predetermined threshold value is 50%. Neither *Swanson*, *Hinton*, nor *Teruyama*, nor any combination thereof, shows or suggests this feature.

VIII. <u>Conclusion</u>

It is respectfully urged that the subject application is patentable over the *Swanson*, *Hinton*, and *Teruyama* references, and any combination thereof, and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

<u>/James O. Skarsten/</u>

James O. Skarsten
Reg. No. 28,346
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants